26th AIAA Applied Aerodynamics Conference
18 - 21 August 2008, Honolulu, Hawaii

AIAA 2008-7177

Parallel Unsteady Overset Mesh Methodology for a Multi-Solver Paradigm with Adaptive Cartesian Grids

Jayanarayanan Sitaraman, * Matt Floros, †
Langley Research Center, Hampton, VA
Andrew M. Wissink,‡ and Mark Potsdam, §
Ames Research Center, Moffett Field, CA

This paper describes a new domain-connectivity module developed to support Chimera-based interfacing of different CFD solvers for performing time-dependent adaptive moving-body calculations of external aerodynamic flows. The domain-connectivity module coordinates the data transfer between different solvers applied in different parts of the computational domain — body fitted structured or unstructured to capture viscous near-wall effects, and Cartesian adaptive mesh refinement to capture effects away from the wall. The CFD solvers and the domain-connectivity module are executed within a Python-based computational infrastructure. The domain-connectivity module is fully parallel and performs all its operations (identification of holes and fringe points, donor cell searches and data interpolation) on the partitioned grid data. In addition, the connectivity procedures are completely automated using the implicit hole-cutting methodology such that no user intervention or explicit hole-map specification is necessary. The capabilities and performance of the package are presented for several test problems, including flow over a NACA 0015 wing, AGARD A2 slotted airfoil, hover simulation of scaled V-22 rotor, and a dynamic simulation of UH-60A rotor in forward flight.

I. Introduction

Traditional CFD codes are often written to support a single gridding paradigm which falls under three main classifications: Cartesian (structured or unstructured), structured-curvilinear (body-fitted) or unstructured (tetrahedral, hexahedral or prismatic). Each meshing paradigm has specific advantages and disadvantages. For example, Cartesian grids are easy to generate, to adapt, and to extend to higher-order spatial accuracy, but they are not well-suited for resolving boundary layers around complex geometries. Structured curvilinear grids work well for resolving boundary layers, but the grid generation process for complex geometries remains tedious and requires considerable user expertise. General unstructured grids are well-suited for complex geometries and are relatively easy to generate, but their spatial accuracy is often limited to second-order, and the associated data structures tend to be less computationally efficient than their structured-grid counterparts. Thus, while a single gridding paradigm brings certain advantages in some portions of the flowfield, it also imposes an undue burden on others. A computational platform that supports multiple mesh paradigms provides the potential for optimizing the gridding strategy on a local basis. However, integrating different meshing paradigms into a single large monolithic code is complex and usually relegates at least one of the models to be less accurate, less optimized, or less flexible than the original standalone solver.

To mitigate the aforementioned issues, in the use of a multiple-mesh strategy within a monolithic code base, we explore a multiple-mesh strategy that is implemented through the use of *multiple* CFD codes, each optimized for a particular mesh type. In particular, we apply unstructured or body-fitted curvilinear grids near the body surface to capture complex geometry and viscous boundary layers, while a short distance

*Research Scientist, NIA, 100 Exploration Way, Hampton VA, jaina@nianet.org, AIAA Member.
†Aerospace Engineer, U.S. Army Research Laboratory., Matthew.W.Floros@nasa.gov, AIAA Member
‡Senior Research Scientist, Scaled Numerical Physics LLC., awissink@mail.arc.nasa.gov, AIAA Member
§Aerospace Engineer, U.S. Army Aeroflightdynamics Directorate (AMRDEC), mpotsdam@mail.arc.nasa.gov, AIAA member

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
AUG 2008		**00-00-2008 to 00-00-2008**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Parallel Unsteady Overset Mesh Methodology for a Multi-Solver Paradigm with Adaptive Cartesian Grids	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **National Aeronautics and Space Administration,Langley Research Center,Hampton,VA,23681**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**22**	

from the body surface, we apply a high-order block-structured adaptive Cartesian solver, that adapts time-dependently to capture wake effects. Previous work [1,2] has addressed the development of a Python-based multi-solver infrastructure using both unstructured (NSU3D[3]) and structured (UMTURNS[4]) near-body solvers and an adaptive Cartesian off-body solver (SAMARC). Figure 1 shows an example calculation of flow over a sphere using this approach for which both the viscous boundary layer and the shed vorticity are accurately captured.

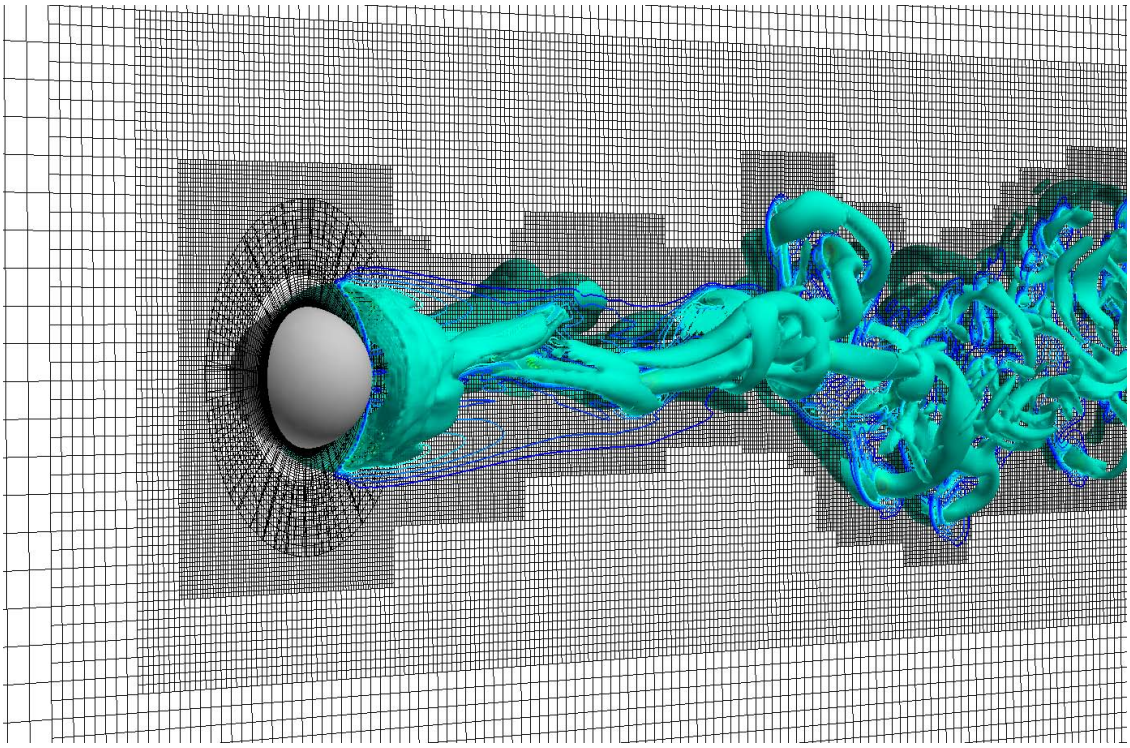


Figure 1. Unsteady flow over a sphere at Re=1000 using the multiple-solver approach. The NSU3D unstructured solver is used near the body surface, the high-order Cartesian AMR solver SAMARC is used in the field, and data is interpolated between the solvers using Chimera-based interpolation.[1]

A critical aspect of the multi-mesh/multi-solver approach is the need for data exchange between the different meshes, which is facilitated in our work using the well-established Chimera-based overset procedure. As shown in Fig 2, the near-body and off-body meshes are constructed to be overlapping and the fringe data are interpolated using a domain-connectivity module. Specifically, the domain-connectivity procedures involve the evaluation of inter-grid boundary data points (points that receive data), donor cells (points that provide data), hole points (points that do not need to be solved) and interpolation weights. The focus of the present paper is on the development of a new domain-connectivity module to support the overset multiple-mesh paradigm efficiently for large-scale unsteady computations with particular application to rotorcraft aerodynamics.

Several Domain-Connectivity approaches have been investigated in the past by various research groups. The prominent among them are PEGASUS5,[5] OVERFLOW-DCF,[6,7] SUGGAR/DiRTlib,[8] CHIMPS,[9] BEGGAR,[10] FASTRAN,[11] and Overture.[12] Evaluation of these packages showed that all of them had certain deficiencies that made them non-ideal for the multi-solver paradigm. For example, PEGASUS5 and SUGGAR while being robust and validated are intended to be integrated with a single solver as opposed to interfacing two solvers, and the domain connectivity operation is not parallel which limits their applicability to large-scale moving body problems. OVERFLOW-DCF proved to be an efficient parallel solver for moving body problems, but it is tightly integrated within the OVERFLOW[7] code, making its applicability to a modular multiple solver paradigm difficult. Also there is no support currently available in the module for unstructured meshes. The CHIMPS package was quite modular with excellent API and had support for parallel execution in moving body environments, but it suffered from lack of support for automated hole-cutting and inefficiency in search procedures. All the above packages required user input in some form or other for

specifying hole-regions (i.e. regions of grid where flow solutions are not performed) and typically performed explicit hole cutting.

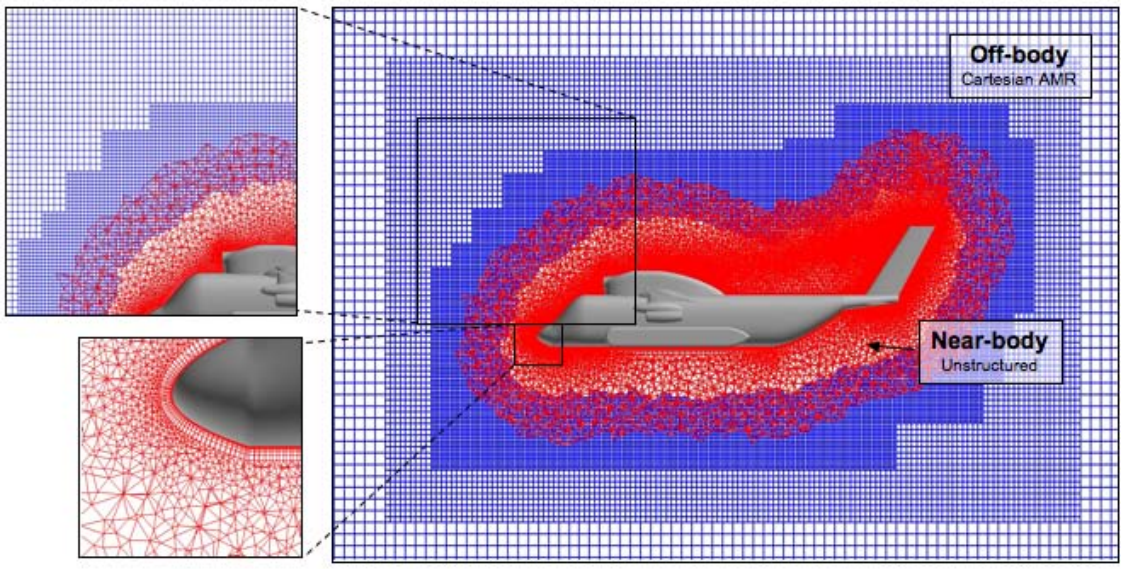


Figure 2. Overset near/off-body gridding. Unstructured or curvilinear grids to capture geometric features and boundary layer near body surface, adaptive block-structured Cartesian grids to capture far-field flow features.

The explicit hole-cutting methodology is known to be prone to several failure modes for moving body problems. An alternate approach is known as "implicit hole-cutting" in which holes and fringe points are determined as part of a search process (i.e. each grid node searches for the best possible donor). This concept was initially investigated by PEGASUS5[5] researchers and eventually deemed too inefficient for dynamic problems. However, recently a research code, NAVAIR-IHC,[13] was able to show efficient usage of this methodology for application to moving body problems. The NAVAIR-IHC code also was evaluated for potential application within the multi-solver paradigm. However, lack of parallelism, implementations that are specific to a structured grid topology, and failure modes for concave grid topologies made it unsuitable for the general application in the multi-solver paradigm with partitioned grid data.

For seamless usage in the computational infrastructure, the Domain-Connectivity software should be modular, parallel, fully-automated, and efficient for unsteady moving body problems.[14] The primary subject of this paper concerns the development and application of such a methodology in the multi-solver context. In addition, this paper will seek to demonstrate the capabilities of the multi-solver paradigm in modeling complex problems with optimal grid distribution and highlight the advantages of adaptive Cartesian grids in resolving flow features.

The new package developed as part of this work is termed PUNDIT, which is an acronym for Parallel Unsteady Domain Information Transfer. In PUNDIT we attempt to develop a package which is modular, parallel and supports efficient automated domain-connectivity operations for all participant grid types (unstructured, structured curvilinear and adaptive Cartesian). To minimize user inputs, we explore the implicit hole-cutting methodology for identification of hole and fringe regions. In addition, PUNDIT will operate in the same computational infrastructure as the participant codes and use data pointers for grids and solution variables directly, thereby promoting ease of interfacing as well as reducing the memory footprint.

II. Methodology

The coupling of the near-body solver, off-body grid manager, off-body solver, and domain-connectivity module are accomplished through a python-based infrastructure with emphasis on preserving modularity of the participating solvers. In addition to the advantages in efficiency and ease in code development, coupling existing mature simulation codes through a common high-level infrastructure provides a natural way to reduce the complexity of the coupling task and to leverage the large amount of verification, validation, and user experience that typically goes into the development of each separate model. The infrastructure discussed

here couples the following codes through a Python infrastructure: (a) the parallel NSU3D code[3] for the unstructured near-body solver, (b) the parallel UMTURNS[4] code as the structured near-body solver, (c) the SAMRAI framework[15] for the off-body Cartesian grid generation, adaptation, and parallel communication, (d) the serial high-order ARC3DC code for solution on the Cartesian blocks, and (e) a domain-connectivity module (PUNDIT).

Brief descriptions of the implementations of the first four components are outlined below. More detailed descriptions of the implementations as well as performance statistics and validation are described in more detail in Refs.[1,2] This paper is devoted to detailed descriptions of algorithms and implementation of the domain-connectivity module (PUNDIT). The infrastructure which integrates the simulation capabilities of all the modules mentioned above is called HELIOS (Helicopter Overset Simulations) and is developed under the DoD HPC HI-ARMS program.

II.A. Python infrastructure

Python-based computational frameworks have been developed previously by several researchers[16] as a means of coupling together existing legacy codes or modules. Such a framework has a number of advantages over a traditional monolithic code structure: (1) it is easier to incorporate well-tested and validated legacy codes rather than to build the capabilities into an entirely new code, (2) there is less code complexity and so maintenance and modification costs are less, and (3) it is easier to test and optimize the performance of each module separately, often yielding better performance for the code as a whole. Essentially, Python enables the legacy solvers to execute independently of one another and reference each other's data without memory copies or file I/O. Further, the Python-wrapped code may be run in parallel using pyMPI or myMPI, with each of the solvers following its native parallel implementations. Further details of the Python infrastructure used here can be found in Wissink et al.[1] and Sitaraman et al.[2]

II.B. Flow Solver Modules

Well-established legacy codes are employed for all of the independent modules in this study. For the unstructured solver in the near-body region, we employ the NSU3D[3] code, which is an implicit node-centered Reynolds-Averaged Navier-Stokes (RANS) code capable of handling arbitrary unstructured mesh elements. In addition, we also use a curvilinear structured mesh solver for the near-body region, namely the UMTURNS[4] code, which is also an implicit RANS solver. For the Cartesian grids in the off-body region, we utilize a Cartesian grid derivative of the well-known ARC3D[17] code, referred to here as ARC3DC. The ARC3DC code employs third-order temporal discretizations using a multi-stage Runge-Kutta time-stepping framework and is capable of up to fifth-order accurate spatial discretizations. Further, the Cartesian grids in the off-body are automatically generated and managed for parallel execution by the SAMRAI infrastructure.[15,18] As mentioned earlier, these codes or modules are combined together using a Python-based framework that orchestrates the execution of these modules and the associated data transfers.

II.C. Meshing paradigm

The meshing paradigm consists of separate near-body and off-body grid systems. The near-body grid typically extends a short distance from the body, sufficient to contain the boundary layer. This grid can be a structured curvilinear grid or an unstructured tetrahedral or prismatic grid that has been extracted from a standard unstructured volume grid or generated directly from a surface triangulation using hyperbolic marching. The reason for using curvilinear or unstructured grids in the near-body region is to properly capture the geometry and viscous boundary layer effects, which are difficult or impossible to capture with Cartesian grids alone. We further note that either structured or unstructured grids will work equally well in the near-body region from the point of view of our infrastructure. Away from the body the near-body grid solution is interpolated onto a Cartesian background mesh with the aid of the domain-connectivity algorithm. This transition normally occurs at a distance wherein the sizing of the near-body grid cells is approximately commensurate with the sizing of the Cartesian mesh in the off-body region.

The off-body grid system consists of a hierarchy of nested refinement levels, generated from coarsest to finest. In the Structured Adaptive Mesh Refinement (SAMR) paradigm,[19] the coarsest level defines the physical extents of the computational domain. Each finer level is formed by selecting cells on the coarser level and then clustering the marked cells together to form the regions that will constitute the new finer

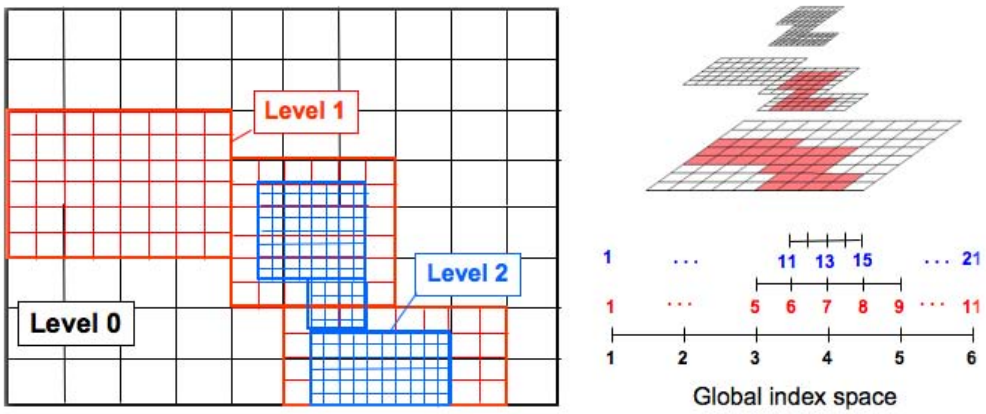


Figure 3. Block structured AMR grid composed of a hierarchy of nested levels of refinement. Each level contains uniformly-spaced logically-rectangular regions, defined over a global index space.

level. Solution-based refinement progresses as follows: physical quantities and gradients are computed at each Cartesian cell using the latest available solution and those cells that hold values deemed to require refinement are marked. The marked cells are then clustered to form the new set of patches that constitute the next finer level. The process is repeated at each new grid level until the geometry and solution features are adequately resolved. We note that this entire procedure is automated within the software and no user intervention is required. The procedure of adaptive mesh refinement is graphically illustrated in Fig. 3.

An example of the overset near-body/off-body meshing strategy is given in Fig. 2, which shows the meshes for flow computations over a helicopter fuselage. Here, the mixed-element unstructured near-body grid envelops the fuselage, while a multi-level Cartesian off-body grid extends from the outer boundary of the near-body grid to the far-field boundary. The two sets of meshes overlap in the so-called fringe region, where data are exchanged between the grids.

II.D. Overset methodology

The overall solution procedure for the overset meshes is as follows. At each iteration step, the solution of the fluid equations in each mesh is obtained independent of each other with the solution in the fringe region being specified by interpolation from the overlapping "donor" mesh as Dirichlet boundary conditions. At the end of the iteration, the fringe data is exchanged between the solvers so that the evolution of the global solution is faithfully represented in the overset methodology. Further, if one or more of the meshes is moving or changing, the fringe regions and the interpolation weights are recalculated at the beginning of the time step. This procedure is repeated for each iteration step until solution convergence is attained in both the near- and off-body meshes.

II.E. Domain-connectivity Module (PUNDIT)

For each solver, the solution at the fringe cells is obtained by interpolation from the overlapping "donor" mesh. The domain connectivity module (PUNDIT) is responsible for determining the appropriate interpolation weights for each fringe point. Further, in the case of multiple overlapping meshes, PUNDIT must also identify one mesh as the donor for each fringe point in each mesh. For static meshes, these operations are done once, at the beginning of the computation, while, for the more general case of moving (or adapting) meshes, the determinations of donors and weights has to be repeated within the time-stepping or iteration loop.

PUNDIT adopts the implicit hole cutting procedure followed by NAVAIR-IHC.[13] The core idea of this approach is to retain the grids with the finest resolution at any location in space as part of the computational domain and interpolate data at all coarser grids in this region from the solution on the fine grid. This results in the automatic generation of optimal holes without any user specification as in the case of explicit hole-cutting. Moreover, the implicit hole-cutting procedure produces arbitrary number of fringe points based on mesh density compared to traditional methods which use fixed fringe width (usually single or double).

The critical parameter that quantifies the quality of a grid cell or node is termed as "resolution capacity". PUNDIT uses the cell volume as the resolution capacity for a grid cell and the average of cell volumes of all associated grid cells as resolution capacity for a grid node. In addition, PUNDIT separates the, near-body to near-body and near-body to off-body domain-connectivity procedures to facilitate automatic off-body grid generation and improve efficiency (Meakin et al.[14]).

Following are the steps followed by PUNDIT to determine the domain connectivity information in a parallel environment. Partitioning of grid and solution data are assumed to be performed using an appropriate solver-based load balancing scheme before these steps are executed.

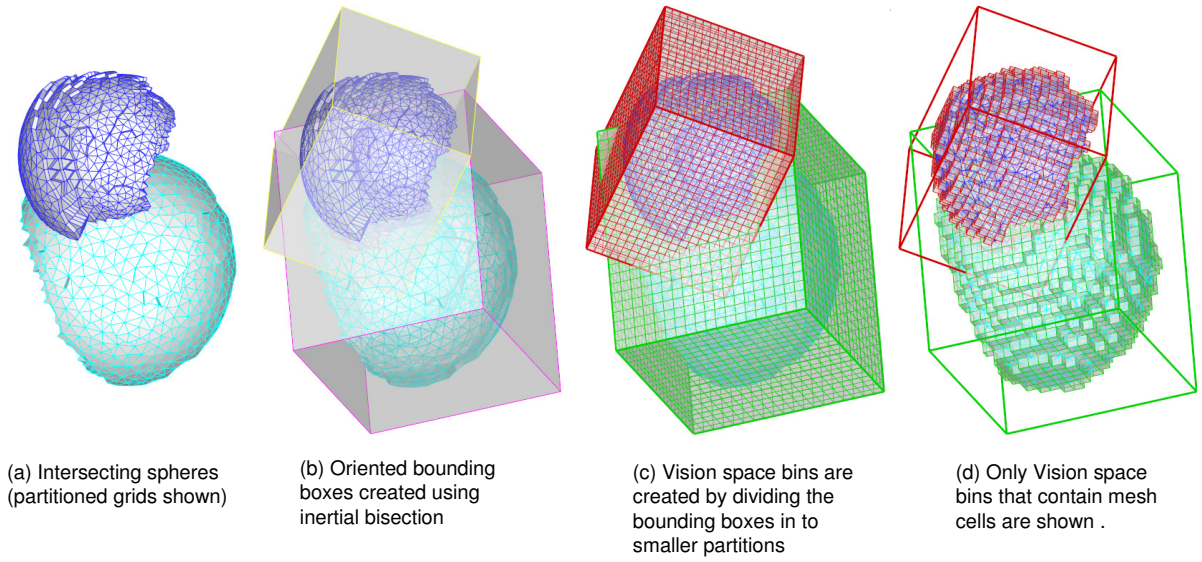


Figure 4. Oriented bounding boxes and vision space bins that are created during preprocessing to accelerate donor search processes.

1. *Registration*: On each processor, the flow solver registers grid and solution data pointers for every grid block (both near body and off-body) with PUNDIT.

2. *Profiling*: PUNDIT profiles each of the grid blocks and forms meta-data representations to facilitate faster donor search operations. The main procedures that are executed in this step are:

   - Minimal *bounding box* computation: Oriented bounding boxes are constructed instead of axis-aligned bounding boxes to minimize the search space (Figure 4(b)).
   - Division of minimal bounding box in to *vision space bins*: The vision space bins are smaller Cartesian boxes within the bounding box. The size of the vision space bins are determined by dividing the volume of the bounding box by the total number of cells contained in it (Fig 4(c) and Fig 4(d)).
   - Generation of a *cumulative fill-table*: The cumulative fill table is a bin-wise reordering of the cell indices that are contained in each vision space bin. They facilitate fast identification of all cells contained within each vision space bin.
   - Estimation of element *resolution capacity*: Element resolution capacity is the grid quality measure for each cell and each grid node. Cell volumes are used as resolution capacity for each grid cell and average of all the associated cell volumes is used as the resolution capacity for each grid node.

3. *Near-body to near-body domain-connectivity*: In this step the domain-connectivity operations are performed between all near-body blocks on all processors. The sequence of the operations is as follows: The bounding boxes constructed in each processor in the previous step are gathered in every processor. Each processor checks for intersection of bounding boxes of its grid blocks with the global set of bounding boxes, which aids in identifying candidate donor grid blocks. The bounding box intersection

checks are optimized such that detection of intersection as well as identification of vision space bins containing potential receiver grid points are simultaneously processed. The identification of vision space bins accelerates the identification of potential receiver grid points as only those nodes which belong in these vision space bins are checked for containment in the donor bounding box.

Following the intersection checks a communication packet is setup and exchanged between all the processors. The communication package consists of a list of coordinates of all potential receiver grid points and their resolution capacity organized such that the donor search can be directed to the appropriate grid block in the candidate processor. Upon completion of this communication, each grid block in each processor will form a list of points for which it needs to locate donor cells. The donor search is then conducted following the algorithm outlined below. Once a list of donors is obtained it is communicated back to the processor that requested the donor search, where an evaluation of potential donor cells which were found is performed, such that the donor of the best resolution capacity is chosen for every receiver grid point. Indices of all the potential donors that are not acceptable are communicated back to the appropriate donor processor such that it can update its donor list. Additionally a quality check is performed to make sure that no donor cells have a receiver point as its vertex. If such a cell is located it is deleted from the donor list and this information is communicated to the receiver processor which adjusts its receiver list accordingly as well. This process rigorously establishes donor quality since all donors will be composed of only nodes which are being solved and are not themselves receiver points. The final product of this step is a communication table consisting of a list of donors and receivers in each processor which is synchronized for data interpolation.

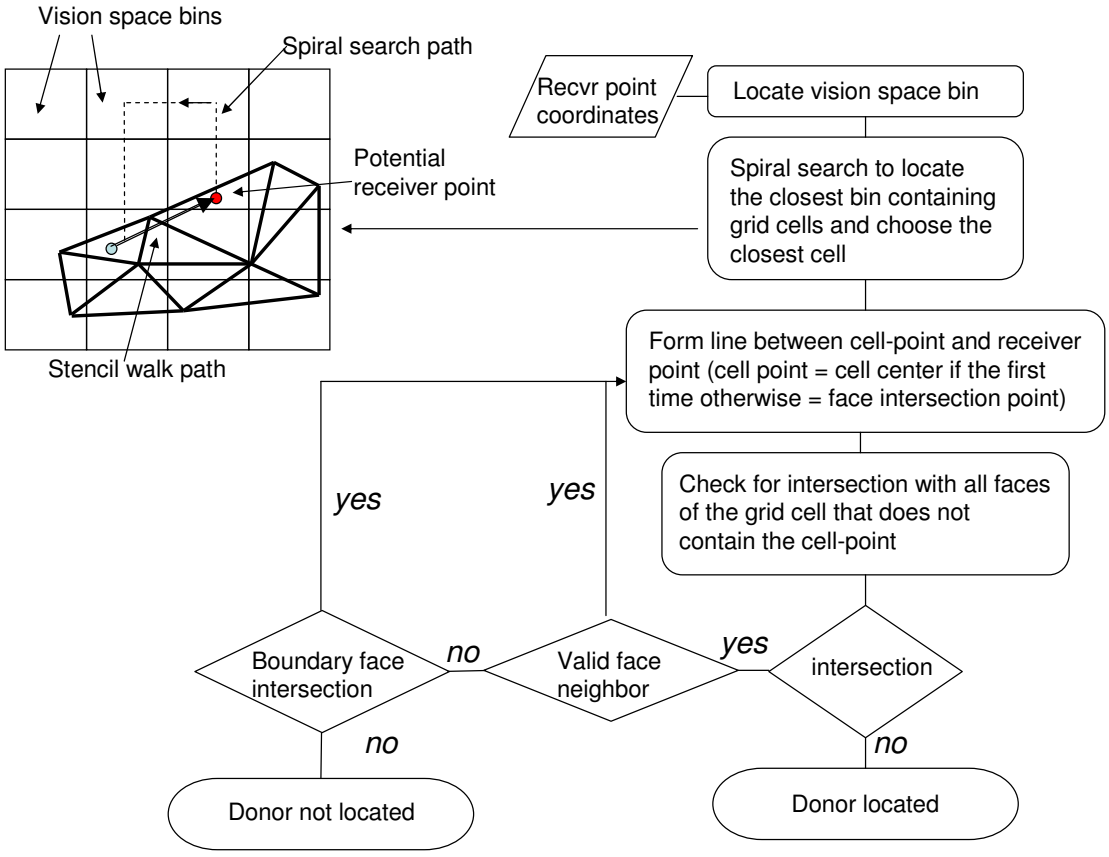


Figure 5. Flowchart of donor finding algorithm that uses vision space bin based localization and stencil-walk based search.

Donor Search Algorithm Figure 5 shows a graphical illustration and flow chart of the donor search algorithm outlined below. The donor search algorithm proceeds as follows: For any potential receiver point, a localization is performed by locating the vision space bin that contains it. This is a trivial

operation since the vision space bins follow a Cartesian structure. Subsequently a spiral search is performed beginning at the vision space bin until a bin with at least one grid cell is located. This grid cell is chosen as the seed for initiating the so called "stencil walk process". A line is created by connecting a point inside a potential donor cell (centroid for the initial cell and face intersection point for subsequent cells) with the receiver point. A check is made to determine if this edge intersects any of the faces of the cell. If an intersection is located, the cell which forms the neighbor at that face is chosen as the next potential donor cell. This procedure continues until a cell with no intersections is located which would be the donor cell for the given point. For a face which shows intersection but has no neighbor (i.e. a boundary face), a check is conducted using the spiral search technique to check if the edge intersects any other boundary faces in the neighborhood. If an intersection is found then stencil walking proceeds to the cell which owns that boundary face. This procedure addresses complex grid boundaries such as those found in partitioned unstructured grids or thin trailing edges. If not the receiver point is pronounced to have no donor cell in the current grid block. In addition, if the final boundary face that the line intersected is a wall boundary face the receiver point will be tagged as a hole-point (i.e. it is inside the solid body).

4. *Off body grid generation*: The adaptive Cartesian mesh generation is dictated by the resolution of the near body meshes at their boundaries. In order to achieve this a list is generated which is composed of all outer boundary points of near body meshes across processors that did not find a near-body donor in the previous step. This list is communicated to the off-body grid manager (SAMRAI) which automatically generates/adjusts nested Cartesian meshes such that the resolution is commensurate at the near-body boundaries.

5. *Off-body to Near-body connectivity*: First valid donors (i.e those with better resolution capacity) for all points in near-body grid blocks are searched in the off-body Cartesian blocks. This search process is extremely efficient because of the isotropic nature of the Cartesian blocks and the global grid indexing followed by the off-body grid manager. A donor Cartesian cell can be located in just a single step. In addition this process identifies all the Cartesian blocks which might contain potential receiver points. Following this valid donors in the near-body grids are searched for all potential off-body grid points using the same process as that was outlined for the near-body to near-body domain-connectivity. Once the donors and receiver points are located for both the off-body and near-body grids a communication pattern is followed which updates the communication tables that synchronize the donor and receiver lists in each processor. Donor quality is again guaranteed by culling out low quality donors and their corresponding receivers from the communication tables. Once the communication tables are finalized the interpolation weights for each grid node of each donor cell in the donor list are computed. Tri-linear basis functions are used for data interpolation and the interpolation weights are computed using a Newton-Raphson procedure.

6. *Data interpolation*: Data is interpolated in a three step process. The first step consists of populating a communication buffer using the solution data and the donor list from the communication tables. These data buffers are exchanged between processors through interprocessor communication in the second step. In the third and final step the solution data is updated in each processor based on the receiver list from the communication tables. In contrast to many of the existing domain-connectivity solvers, PUNDIT minimizes the communication overhead by maintaining the interpolation weights in the host processor. The client processor only receives solution data and is capable of assigning them appropriately because of the synchronization in communication tables. Moreover interpolation weights are evaluated only for the final list of donors, minimizing the amount of floating point operations. In order to support interfacing of solvers which follow different non-dimensionalizations for flow variables, PUNDIT requests non-dimensionalizing factors for each flow variable from each participant solver. The data buffers that are exchanged are in the dimensional form and appropriate non-dimensionalization is applied at the update step based on the factors provided by each participant code. It is worth noting that the implementation of data interpolation is quite general and does not impose any restriction on the type or number of flow variables.

III. Software verification

III.A. Interpolation Accuracy

All the polyhedra that are common in unstructured mixed-element meshes (tetrahedra, pyramids, prisms and hexahedra) are supported in PUNDIT. Tri-linear basis functions are used to perform data interpolation inside each of these polyhedra. The accuracy of interpolation was verified by constructing unstructured meshes in a unit cube of incremental resolutions. PUNDIT is used then to perform search and interpolation on a random set of points distributed in the unit cube using the data at the grid nodes (which are prescribed using chosen test functions) of the unstructured meshes. The interpolated values are compared with exact test function values to estimate the interpolation error. Figure 6 shows the variation of interpolation error with improving resolution. For linear test functions, the tri-linear interpolation gives exact solutions within machine precision. For a non-linear test functions interpolations in all types of polyhedra show 2nd order accuracy. Interpolation accuracy is also verified to be 2nd order for a test function that uses the Lamb-vortex solution.

III.B. Scalability studies for UH-60A and TRAM rotors

The performance of the domain-connectivity module is studied by varying the number of processors used for program execution. Note that partitioning of the grid data is handled by flow solvers and PUNDIT operates on this partitioned grid data. Figure 7 shows the speedup in wall-clock time with increasing number of processors. Three sets of grid systems are studied to calibrate scalability. The TRAM grid system consists of 0.8 million near body nodes and 5.9 million off body nodes. The coarse UH-60A grid system is composed of 1 million near-body nodes and 11 million off-body nodes. The fine UH-60A grid system contains 2 million near-body nodes and 30 million off-body nodes. More details on these test cases are described in the application results section (IV.E and IV.D). Good linear scalability was observed for all three cases up to 12 processors. Beyond that the scalability drops off depending on grid size. Larger mesh systems such as the fine grid for UH-60A begin to drop off at about 32 processors, while the TRAM rotor starts to drop much earlier at about 12 processors. The reason for scalability drop-off is the load-imbalance created by non-optimality of solver based partitioning for domain-connectivity procedures. Redistribution of the computational work for searching donors evenly after an initial evaluation of donor processors could be an approach to mitigate this problem (For e.g. the "rendezvous" approach described in Plimpton et al[20]).

IV. Application results

We present results using PUNDIT within the flow solver package HELIOS for several examples. The results section is organized as follows: first the NACA0015 fixed wing test case is presented which demonstrates the advantages of adaptive grids and higher-order methods. The next set of results illustrates the capability of PUNDIT to perform domain-connectivity operations which automate hole cutting and generate optimal overlaps. The domain-connectivity solutions are shown for the AGARD A2 test case (NHLP-2D slotted airfoil) and the V-22 wing-nacelle combination. Following that we present results for rotorcraft simulations for the 1/4th scaled V-22 (TRAM) and the UH-60A rotors.

IV.A. Flow over NACA0015 wing

The multi-code approach is used to study the flow-field over a NACA 0015 rectangular wing. The primary objective of this computation is to validate the multi-code simulation capability for a high-Reynolds number turbulent external flow problem. The experimental data used for validation are those measured by McAlister et al.[21] The wing geometry is a rectangular planform with a square-tip and has an aspect ratio of 3.3. The test point for which validation results are presented is at the following operating condition: $Mach$=0.1235, Re=1.5e6, α=12^o. For this case, computations are performed in the near-body region using the UMTURNS flow solver using the structured curvilinear meshes. The wing is modeled in the full-span configuration. The Spalart-Allmaras turbulence model is used for RANS closure.

Figure 8(a) shows the domain-connectivity solution obtained for the NACA0015 grids. Sections of the grid system in the X-Y plane and X-Z plane are shown. The implicit domain connectivity algorithm employed not only produces optimal overlap but also ensures commensurate grid spacing of off-body and near-body

computational domains at regions of overlap. The pressure distributions obtained from the predictions are compared with measured experimental data in Figure 8(b). Fair agreement can be noted between the measurement and prediction at all the span locations, although the leading edge suction peak is consistently under-predicted, This may be attributed to wind-tunnel wall effects which are not modeled in the analysis presently.

Figure 9 shows the evolution of the vortex structures from the tip of the wing. The flowfield is dominated by the presence of the vortex sheet which rolls up into a strong tip vortex quite close to the wing. This tip vortex remains coherent up to about 15 chords behind the wing and then begins to break down in to smaller structures. Vorticity magnitude is used as the mesh adaptation criterion for the off-body solver. The meshes represented in the figure show four levels of adaption. It can be observed that the mesh system is able to track the vorticity and automatically adapt to regions of high vorticity (around the area of tip vortices). The figure also shows comparison of swirl velocities predicted at the tip-vortex locations with measured experimental data. The peak magnitudes are under predicted by about 20 percent which is consistent with the under-prediction of the leading edge suction peaks. However, the vortex shows minimal diffusion because of numerical dissipation. The use of a 5th-order accurate numerical scheme (6th-order central difference with 5th-order dissipation terms) in the off-body solver in conjunction with adaptive Cartesian grids facilitates the improved resolution of the tip vortex structure.

Figure 10 further elucidates the advantages of the higher-order scheme combined with solution-based mesh adaptation. The swirl velocities near the vortex location are presented for four different solution algorithms and compared with the test data. From the vorticity contour plots, it can be noted that the isotropic off-body grids are better able to preserve the coherence of the vortex better than stretched grids that are typical of structured near-body meshes. Also, improvements can be noted both in the vorticity contour magnitudes as well as swirl velocity profiles with the inclusion of mesh adaptation and use of 5th-order accurate numerical schemes.

IV.B. AGARD A2 slotted airfoil test case

This section illustrates the capabilities of PUNDIT for the AGARD A2 (NHLP 2D slotted airfoil) test case. Although topologically simple, this problem is challenging for computation of domain-connectivity because of multiple overlapping meshes in close proximity. This slotted airfoil is one of the well established cases for validation of CFD codes.[22,23]

The experimental data for this case are obtained from the electronic supplement of the AGARD 303 report.[24,25] The surface pressure distributions obtained by the computations and their comparison with the experimental data is shown in Figure 11. Excellent agreement can be noted in the suction peaks and chordwise variation of pressure coefficients for all the elements of the slotted airfoil. This validation study provides confidence in the accuracy of the domain-connectivity module.

Figure 12(a) shows the original overlapping meshes that describe the geometry of the slotted airfoil problem. Figures 12(b) and 12(c) show the detail of the leading and trailing edge section which contain regions which have three overlapping meshes. Finally Figures 12(d) and 12(e) show the results after performing the domain-connectivity operation. In these figures only the solver points are shown. It can be noticed that at the solver point boundaries the grid resolutions of all overlapping meshes are commensurate with each other which improves the quality of data interpolation. In addition the amount of overlap is optimized, which in turn minimizes the amount of double solving and redundancy in the flow solution.

Figure 12(e) shows the contours of stream-wise momentum near the leading edge slat and trailing edge flap. The continuity of the contours across the mesh boundaries illustrates the quality of data interpolation achieved by maintaining optimal connectivity. Data interpolation between mesh cells and nodes of differing resolution capacity often causes large unphysical oscillations in contours of flow variables at grid interfaces.

IV.C. V-22 Wing-nacelle test case

The wing-pylon test case is one of the test cases which was originally used for benchmarking various domain-connectivity software to evaluate their usability and performance withing the multi-solver paradigm. Therefore, PUNDIT was applied to this problem to evaluate its performance against existing technology. All results were computed on an AMD64 single-node work station.

The grid system contains 6.7 million grid points distributed among 22 structured grid blocks. The other domain-connectivity software that were tested for this case are SUGGAR, CHIMPS, OVERFLOW-DCF and

NAVAIR-IHC. OVERFLOW-DCF was tested using both explicit hole cutting (object X-rays) and implicit hole cutting. SUGGAR was tested using the explicit hole cutting (binary trees). CHIMPS has no capability for explicit-hole cutting and was tested in the implicit hole cutting mode. NAVAIR-IHC used implicit hole cutting similar to PUNDIT.

The wall-clock time taken by PUNDIT for the full calculation of domain connectivity was 18.5 seconds which was 6X faster than CHIMPS, 3X faster than OVERFLOW-DCF with implicit hole cutting and 2X faster than SUGGAR with explicit hole cutting. However PUNDIT was still 4X slower than NAVAIR-IHC with implicit hole-cutting and 1.5X slower than OVERFLOW-2DC with explicit hole-cutting. The timings for explicit hole cutting methods include the time required to generate hole specification (x-rays, trees). For problems with rigid body motions this may be a one time cost, although deformable bodies do require regeneration.

Figure 13 shows the grid system and the domain connectivity solution obtained for the V-22 wing-nacelle problem. Representation of the geometric data was quite cumbersome because of the complexity of this test case. Therefore subsets of grid system where there are multiple overlapping meshes are shown. The figures show the grid system before and after connectivity. Notice that optimal overlaps and hole cuts are achieved for this case similar to the AGARD-A2 case described earlier.

IV.D. Hover simulation of TRAM rotor

Hover simulation for the TRAM (1/4 scaled V-22) rotor was performed using the HELIOS infrastructure with PUNDIT for grid connectivity. Figure 14 illustrates the mesh system, connectivity solution, and flow solution obtained for this test case. This test case also shows the capability of PUNDIT to handle multiple unstructured meshes. The near-body solver used for this simulation is the unstructured NSU3D code.

The TRAM mesh system consists of three unstructured grid blocks generated by rotating and combining a single blade unstructured mesh. PUNDIT performs domain-connectivity between these grid blocks first and determines the resolution and geometry requirements for the off-body meshes. The off-body mesh generation is performed subsequently such that the resolution of the grid cells near the outer boundaries of near-body meshes is commensurate with the off-body grid cells in that region. Consequently PUNDIT performs the domain-connectivity solution between the off-body and near-body systems and determines the off-body and near-body computational domains. The off-body computational domains are shown overlapped with near-body meshes to illustrate the extent of overlap created. In addition the off-body domains are shown by themselves to illustrate the hole geometry created by the implicit hole-cutting procedure. Overall, it can be observed that PUNDIT is capable of producing optimal domain-connectivity solutions for the mesh system used in this case, even at the center of the rotor where there are multiple overlap regions. It is worth noting that an explicit-hole cutting methodology needs sufficient user intervention to make the appropriate hole cuttings for a mesh system such as this, especially at the overlap regions.

The flow solution obtained from the hover simulation is also shown in the figure which shows the dynamic adaption of the off-body grid system to the tip vortex structure. The solution slice shows contours of z-momentum superimposed on the grid system. Large gradients of z-momentum indicate a larger presence of vorticity as observed in the tip regions where the mesh density is automatically increased. Note that domain-connectivity is performed on the fly in this simulation after each off-body grid adaptation step.

IV.E. Moving mesh computations for rotor aeromechanics

The last case shown is the moving mesh problem for the UH-60A helicopter in forward flight. The meshes used for this problem are shown Figure 15. Figure 15(a) shows the grid system with a curvilinear C-O type topology (near-body) overlapped by a Cartesian mesh (off-body). The flow solutions for the near body grids are performed by the UMTURNS code and those for the off-body grids are performed by the SAMARC code. Figures 15(b) and 15(c) show sections along the z=0 plane and $x = 0$ plane for the entire grid system illustrating the computational domains (solver points) in the near- and off-body regions. The problem includes elasticity as well as rotation of the near-body meshes around the z axis. Prescribed aero-elastic deformations which are obtained from previous CFD/CSD computations[27] are used for verification of the methodology. Since the near-body meshes are moving, the domain-connectivity has to be performed at every time step. The domain-connectivity procedures required about 10 percent of the total time taken for a time step (computations were conducted on 32 processors, which is the scalability limit of PUNDIT for this problem).

Figure 15(d) shows the contours of the z-momentum (downwash) for the flow solution obtained. The vorticity shed from the rotor blades induces the downwash observed in the figure. During the advancing azimuth the aerodynamic loading towards the tip of the blades becomes negative causing them to produce much less downwash compared to the retreating side where the aerodynamic loading is positive. The downwash distribution is also an indicator of the vortex wake geometry shed form the rotor blade; large gradients indicate the presence of strong vortex structures. A comparison of aerodynamic loading (sectional lift and pitching moment at 86 percent radial station) is shown in Figure 15(e) which compare favorably with the measured flight test data.[26]

V. Summary and Conclusions

We have demonstrated the potential of the multi-solver paradigm and automated domain-connectivity in resolving flow physics in unsteady problems which involve vortex dominated flow fields as well as moving bodies. The domain-connectivity module (PUNDIT) shows encouraging trends in obtaining optimal domain-connectivity, improving interpolation accuracy and automated parallel execution. The improvements that PUNDIT brings over the existing domain connectivity technology can be categorized into three areas:

1. *Automation*: There is no user input or intervention required for PUNDIT, which makes it completely automated and shows potential for large productivity gains. Existing technologies often require high-levels of user expertise because of the need to perform explicit hole cutting.

2. *Scalability*: None of the existing technologies provide scalable execution for all the domain connectivity procedures. We have demonstrated linear-scalability of execution if at least 1 million points are available per processor. However, this is still quite short of meeting future production demands of highly-scalable execution. The main reason for the drop-off in scalability is because of solver-based mesh-partitioning, which is non-optimal for domain connectivity. This is because only a subset of processors will have intensive domain connectivity operations as the grid system becomes more finely partitioned, leaving the entire procedure unbalanced. Algorithms for redistribution of search load on the fly are envisioned to mitigate the load imbalance.

3. *Arbitrary element types and relative motion capability*: PUNDIT supports multiple solvers (both intra-code and inter-code), multiple mesh types and performs all the domain connectivity operations in a distributed computing platform for moving body problems. Existing technologies do support many of these features but none of them support all of the features together.

Following are the summaries of the specific observations that pertain to the various application test cases that were studied:

1. The NACA0015 test case shows the capability of adaptive meshes and high-order solvers to preserve the coherence of vortex structures. Comparison with experimental data shows underprediction of peak swirl velocity magnitudes. However these are attributed to the lift deficiency observed from the pressure distributions. Comparison with experimental data is likely to improve with the inclusion of wind-tunnel wall modeling.

2. The AGARD A2 airfoil-flap-slat test case shows the ability of PUNDIT to obtain optimal domain-connectivity and facilitate smooth solution interpolation. Excellent correlation obtained with experimental data for the pressure distributions provides validation for the methodology.

3. Performance studies for the wing-nacelle test case indicated improved performance from the existing mature technologies (OVERFLOW, SUGGAR, CHIMPS). However the performance is still lower than other research technologies (NAVAIR-IHC). Closer analysis of the search algorithms and coordination with NAVAIR-IHC developer should improve efficiency.

4. Studies for the TRAM rotor show the capability of automated mesh adaptation and domain-connectivity solution for a rotorcraft problem in a parallel processing environment

5. Application to the aeroelastic simulation of a UH-60A helicopter further illustrates the capability of dynamic domain-connectivity management in a parallel processing environment with reasonable cost.

VI. Acknowledgements

Development was performed at the HPC Institute for Advanced Rotorcraft Modeling and Simulation (HIARMS) located at the US Army Aeroflightdynamics Directorate at Moffett Field, CA, which is supported by the Department of Defenses High Performance Computing Modernization Office (HPCMO). The authors would like to acknowledge Dr. Robert Meakin for many of the original ideas and critical concepts that have been used in the development of the domain-connectivity module. We also gratefully acknowledge the contributions of Dr. Venke Sankaran in reviewing the manuscript, Dr. Willam Chan for providing test cases and insights, Dr. Buvana Jayaraman for verifying the robustness of the simulation capability and Dr. Beatrice Roget for discussions on search algorithms. We thank Dr. Roger Strawn and Dr. Christopher Atwood for providing the leadership and encouragement. In addition, we are grateful to Dr. Thomas Pulliam for the development of the higher-order Cartesian-mesh solver (ARC3DC). We also thank Dr. Dimitri Mavriplis and Dr. Zhi Yang at the University of Wyoming for providing the NSU3D code. Finally, we are thankful to Dr. Edwin van Der Weide and Prof. Juan Alonso at Stanford University for the use of the CHIMPS interpolation package, Dr. Ralph Noack at Applied Research Laboratory for the usage of SUGGAR module and Yikloon Lee at NAVAIR for the usage of the NAVAIR-IHC code.

References

[1]Wissink, A., Sitaraman, J., Mavriplis, D., Pulliam, T., and Sankaran, V., "A Python-Based Infrastructure for Overset CFD with Adaptive Cartesian Grids," *46th AIAA Aerospace Sciences Meeting*, AIAA, Washington, DC, January 2006.

[2]Sitaraman, J., Katz, A., Jayaraman, B., Wissink, A., and Sankaran, V., "Evaluation of a Multi-Solver Paradigm for CFD using Overset Unstructured and Structured Adaptive Cartesian Grids," *AIAA Aerospace Sciences Meeting*, AIAA, Washington, DC, January 2008.

[3]Yang, Z. and Mavriplis, D., "Higher-Order Time Integration Schemes for Aeroelastic Applications on Unstructured Meshes," *44th AIAA Aerospace Sciences Meeting*, AIAA, Washington, DC, January 2006.

[4]Sitaraman, J. and Baeder, J. D., "Evaluation of the Wake Prediction Methodologies used in CFD Based Rotor Airload Computations," *24th AIAA Conference on Applied Aerodynamics*, AIAA, Washington, DC, June 2006.

[5]Rogers, S. E., Suhs, N. E., and Dietz, W. E., "PEGASUS 5: An Automated Preprocessor for Overset-Grid Computational Fluid Dynamics," *AIAA Journal*, Vol. 41, No. 6, June 2003.

[6]Meakin, R. L., "Object X-Rays for Cutting Holes in Composite Overset Structured Grids," *15th AIAA Computational Fluid Dynamics Conference*, AIAA, Washington, DC, June 2001.

[7]Buning, P. G. et al., *OVERFLOW Users Manual*, NASA Langley Research Center, July 2003.

[8]Noack, R. W., "SUGGAR: A General Capability for Moving Body Overset Grid Assembly," *17th AIAA Computational Fluid Dynamics Conference*, AIAA, Washington, DC, June 2005.

[9]Alonso, J. J., Hahn, S., Ham, F., Herrmann, M., Iaccarino, G., Kalitzin, G., LeGresley, P., Mattsson, K., Medic, G., Moin, P., Pitsch, H., Schlüter, J., Svärd, M., Van der Weide, E., You, D., and Wu, X., "CHIMPS: A High-Performance Scalable Module for Multi-Physics Simulations," *42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, AIAA, Washington, DC, July 2006.

[10]Belk, D. M. and Maple, R. C., "Automated Assembly of Structured Grids for Moving Body Problems," *12th AIAA Computational Fluid Dynamics Conference, Part 1*, AIAA, Washington, DC, June 1995, pp. 381–390.

[11]Wang, Z. J., Parthasarathy, V., and Hariharan, N., "A Fully Automated Chimera Methodology for Multiple Moving Body Problems," *36th AIAA Aerospace Sciences Meeting*, AIAA, Washington, DC, January 1998.

[12]David L. Brown, W. D. H. and Quinlan, D. J., "Overture: Object-Oriented Tools for Overset Grid Applications," *17th AIAA Conference on Applied Aerodynamics*, AIAA, Washington, DC, June 1999.

[13]Lee, Y. and Baeder, J. D., "Implicit Hole Cutting—A New Approach to Overset Grid Connectivity," *16th AIAA Computational Fluid Dynamics Conference*, AIAA, Washington, DC, June 2003.

[14]Meakin, R. L., Wissink, A. M., Chan, W. C., Pandya, S. A., and Sitaraman, J., "On Strand Grids for Complex Flows," *18th AIAA Computational Fluid Dynamics Conference*, Washington, DC, June 2007.

[15]Hornung, R. D., Wissink, A. M., and Kohn, S. R., "Managing Complex Data and Geometry in Parallel Structured AMR Applications," *Engineering with Computers*, Vol. 22, No. 3–4, December 2006, pp. 181–195, See `www.llnl.gov/casc/samrai`.

[16]Alonso, J. J., LeGresley, P., and Van Der Weide, E., "pyMDO: A Framework for High-Fidelity Multi-Disciplinary Optimization," *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, AIAA, Washington, DC, August 2004.

[17]Pulliam, T. H., "Euler and Thin-Layer Navier-Stokes Codes: ARC2D, and ARC3D," *Computational Fluid Dynamics Users Workshop*, March 1984.

[18]Hornung, R. D. and Kohn, S. R., "Managing Application Complexity in the SAMRAI Object-Oriented Framework," *Concurrency and Computation: Practise and Experience*, Vol. 14, 2002, pp. 347–368.

[19]Berger, M. J. and Colella, P., "Local Adaptive Mesh Refinement for Shock Hydrodynamics," *Journal of Computational Physics*, Vol. 82, 1989, pp. 65–84.

[20]Plimpton, S. J., Hendrickson, B., and Stewart, J. R., "A parallel rendezvous algorithm for interpolation between multiple grids," *Journal of Parallel Distributed Computing*, Vol. 64, 2004, pp. 266-1276.

[21]McAlister, K. W. and Takahashi, R. K., "NACA 0015 Wing Pressure and Trailing Vortex Measurements," NASA TP 3151, Ames Research Center, November 1991.

[22]Elsenaar, A. et al., "A Selection of Experimental Test Cases for the Validation of CFD Codes Volume 1," AGARD AR 303, Advisory Group for Aerospace Research & Development (AGARD), 7 Rue Ancelle, 92200 Neuilly-Sur-Seine, France, August 1994.

[23]Burt, M., "Summaries of the Test Cases," *A Selection of Experimental Test Cases for the Validation of CFD Codes Volume 1*,[22] Chap. 5, pp. 55–133.

[24]Advisory Group for Aerospace Research & Development (AGARD), "A Selection of Experimental Test Cases for the Validation of CFD Codes Supplement: Datasets A–E," Electronic Data, August 1994, AGARD-AR-303-SUPPL.

[25]Moir, I. R. M., "Measurements on a Two-Dimensional Aerofoil with High-Lift Devices," *A Selection of Experimental Test Cases for the Validation of CFD Codes Volume 2*, Advisory Group for Aerospace Research & Development (AGARD), 7 Rue Ancelle, 92200 Neuilly-Sur-Seine, France, August 1994, pp. A2–1–A2–12.

[26]Kufeld, R. M., Balough, D. L., Cross, J. L., Studebaker, K. F., Jennison, C. D., and Bousman, W. G., "Flight Testing the UH-60A Airolads Aircraft," *American Helicopter Society 51st Annual Forum*, May 1994, pp. 557–577.

[27]Potsdam, M., Yeo, H., and Johnson, W., "Rotor Airloads Prediction Using Loose Aerodynamic/Structural Coupling," *Journal of Aircraft*, Vol. 43, No. 3, 2006, pp. 732–742.

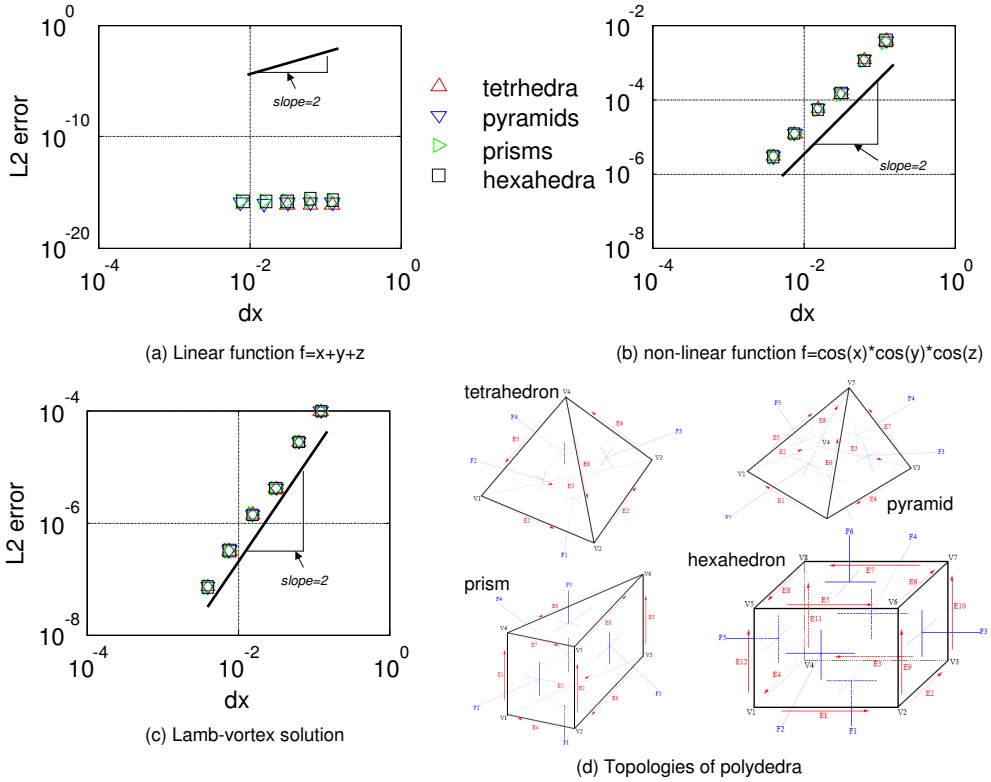


Figure 6. Variation of interpolation errors for polyhedra shown in subplot (d): subplot(a) shows error variation for a linear test function, subplot(b) shows error variation for a non-linear test function and subplot(c) shows error variation for the Lamb-vortex solution.

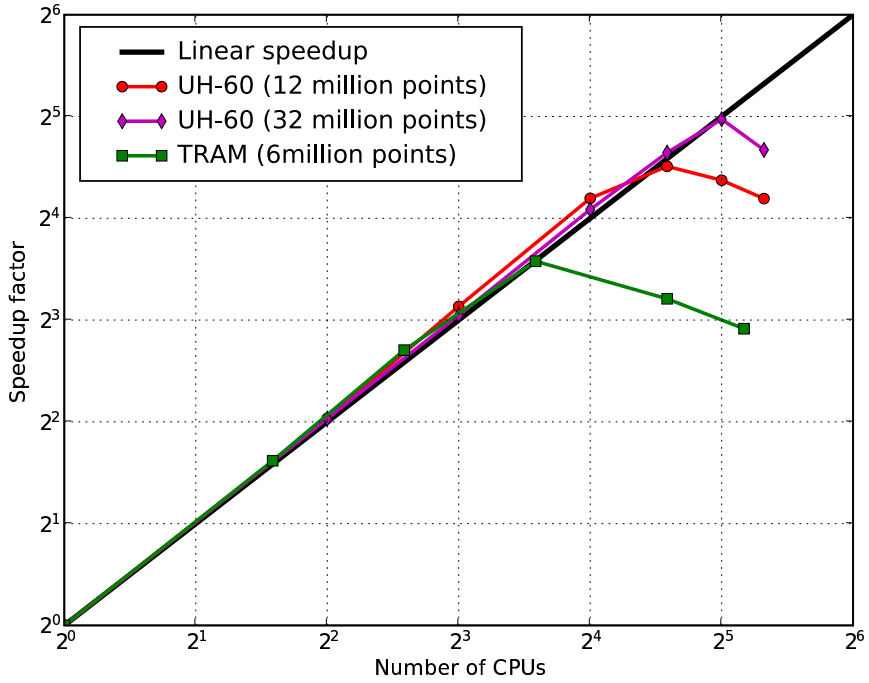
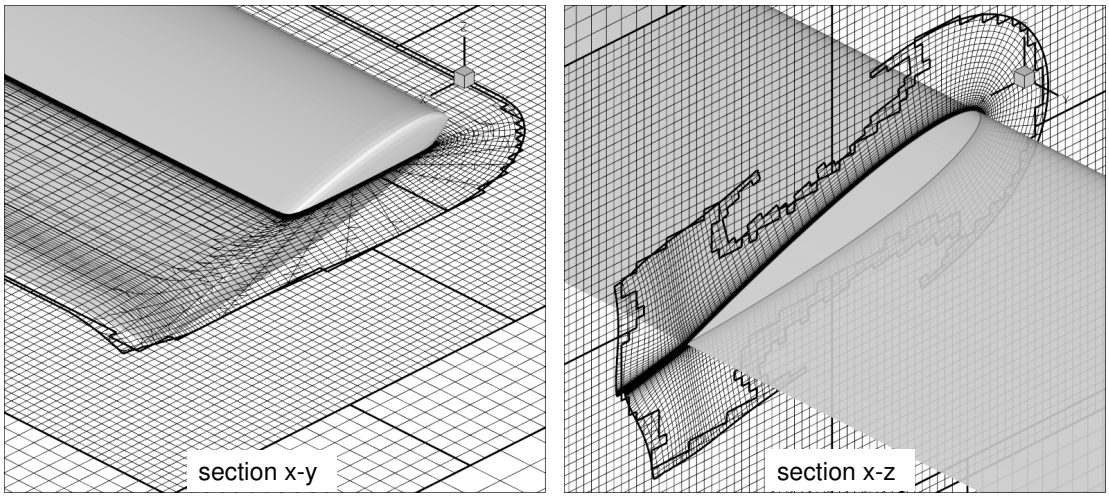
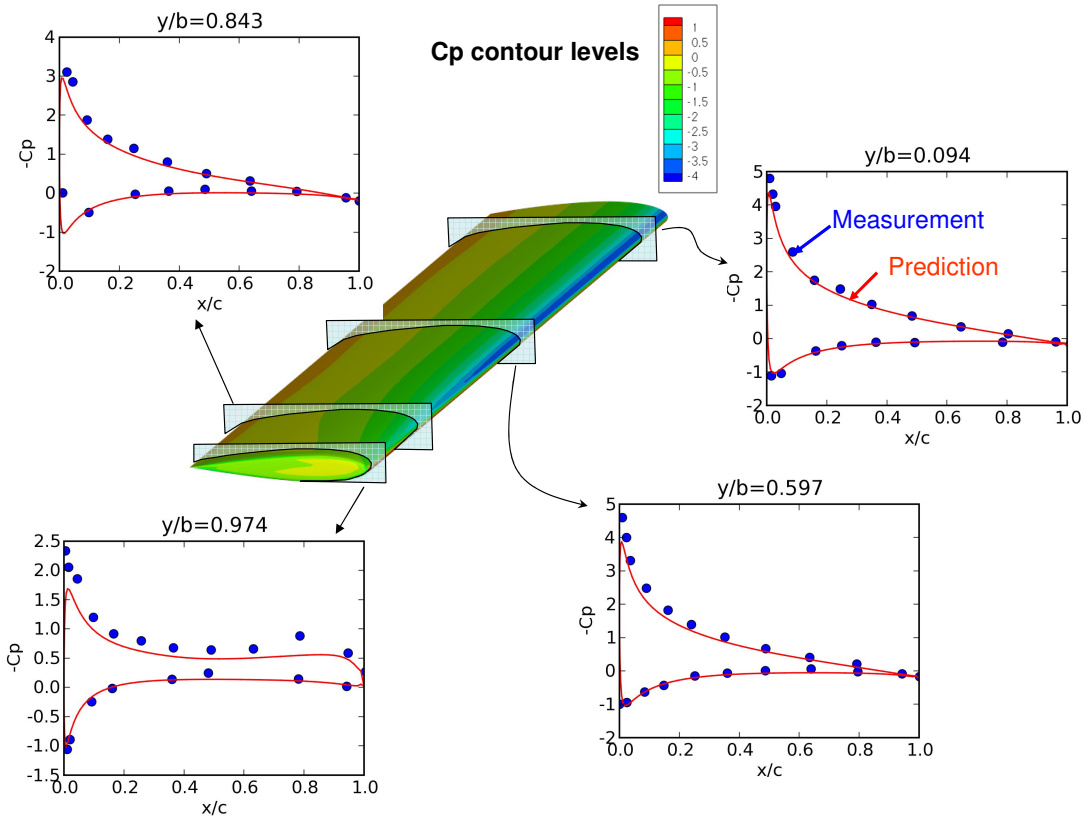


Figure 7. Scalability of PUNDIT with increasing number of processors for TRAM and UH-60A datasets.



(a) NACA 0015 off-body and near body grids after domain connectivity operation (plots shows only those parts of grid which perform flow solution)



(b) Validation of predicted pressure distributions with experimental data (McAlister et. al)

Figure 8. Prediction of steady flow over NACA0015 wing at M=0.1235, α=12 deg and Re=1.5e6 using HELIOS. domain-connectivity solutions are shown in subfigure(a) and comparision of pressure distributions are shown in subfigure(b)

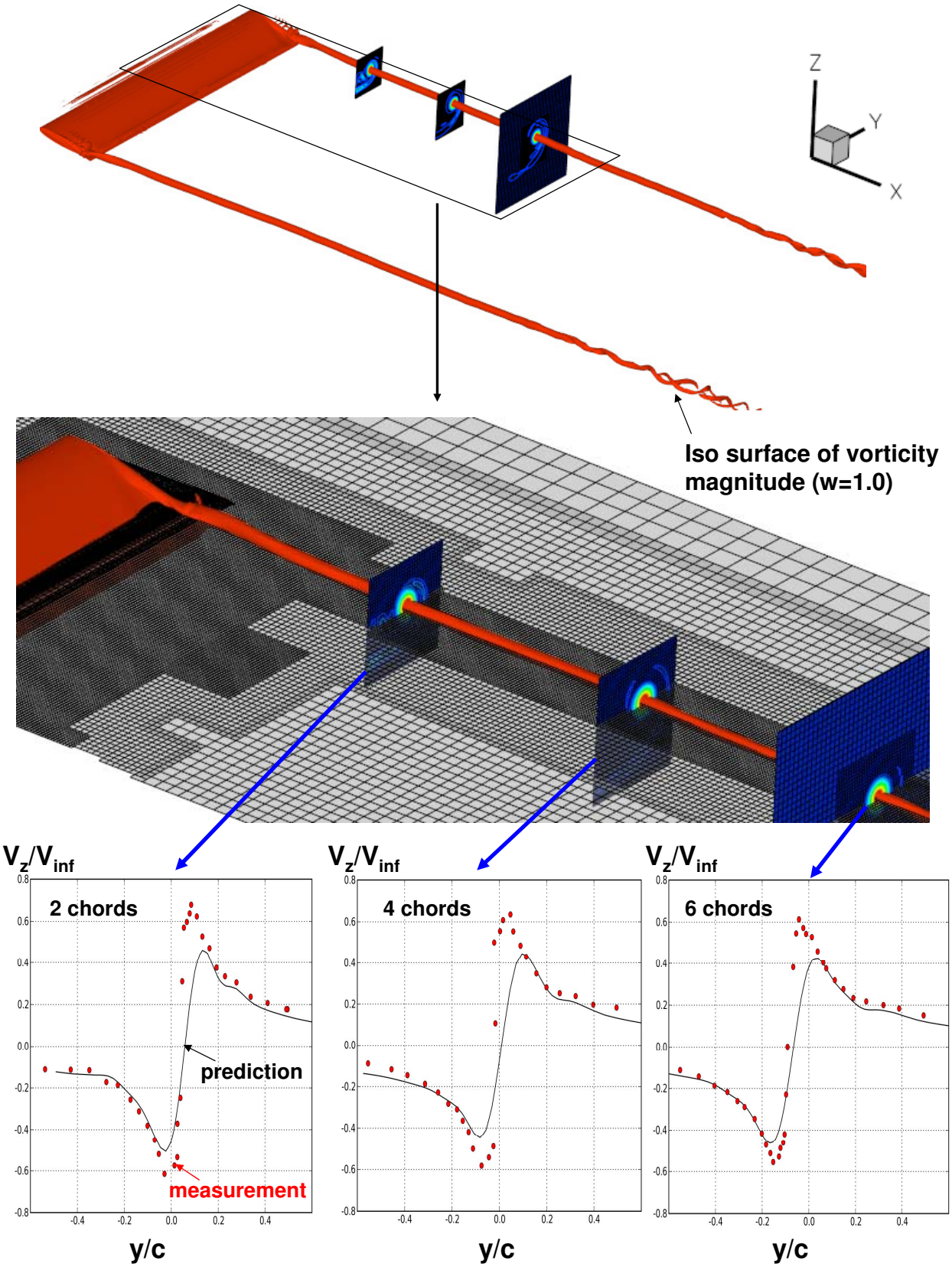


Figure 9. Prediction of steady flow over NACA0015 wing at M=0.1235, α=12 deg and Re=1.5e6 using HELIOS; Predicted tip-vortex trajectory and comparison of predicted swirl velocities with experimental data.

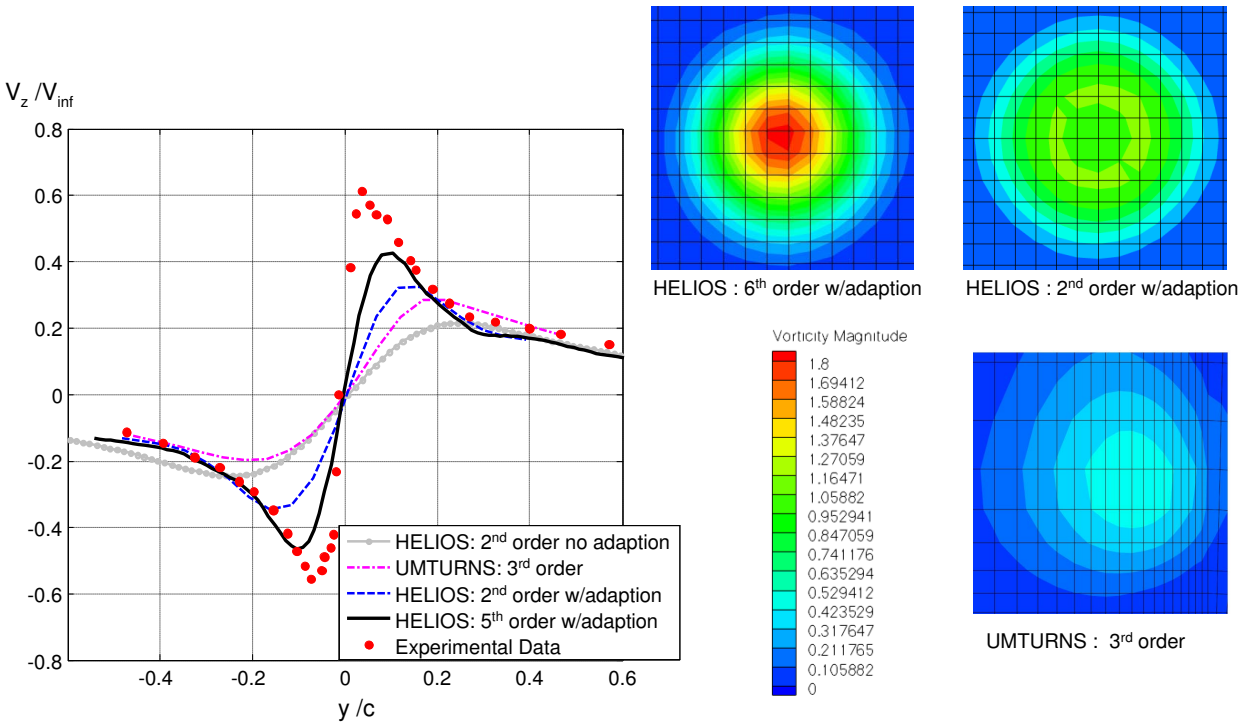


Figure 10. Prediction of steady flow over NACA0015 wing at M=0.1235, α=12 deg and Re=1.5e6 using HELIOS; Comparision of predicted swirl velocity profiles and vorticity contours for different numerical solution strategies.

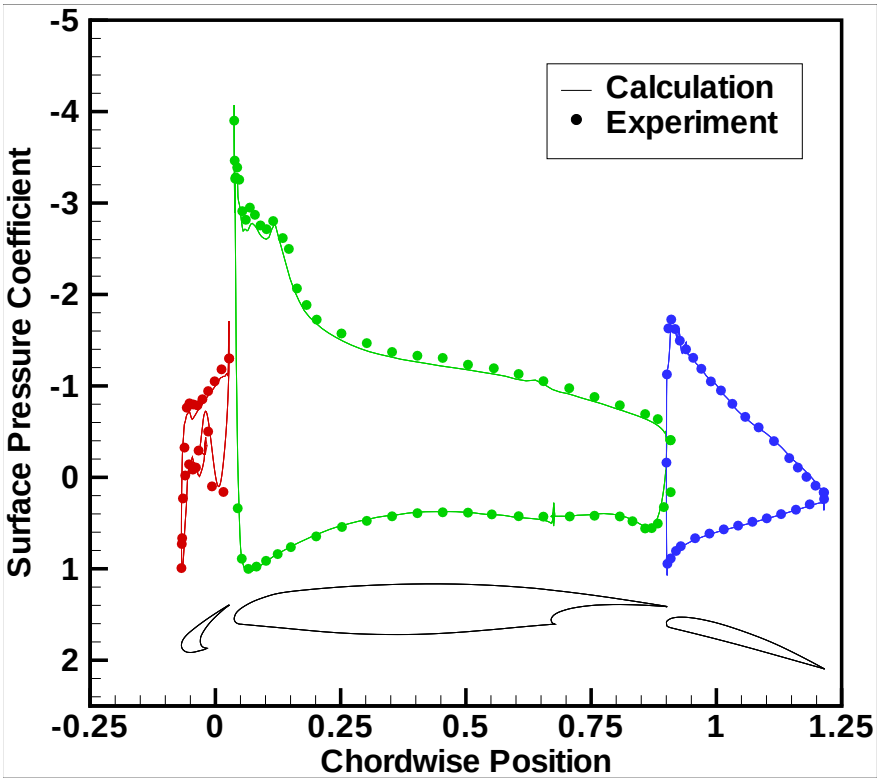
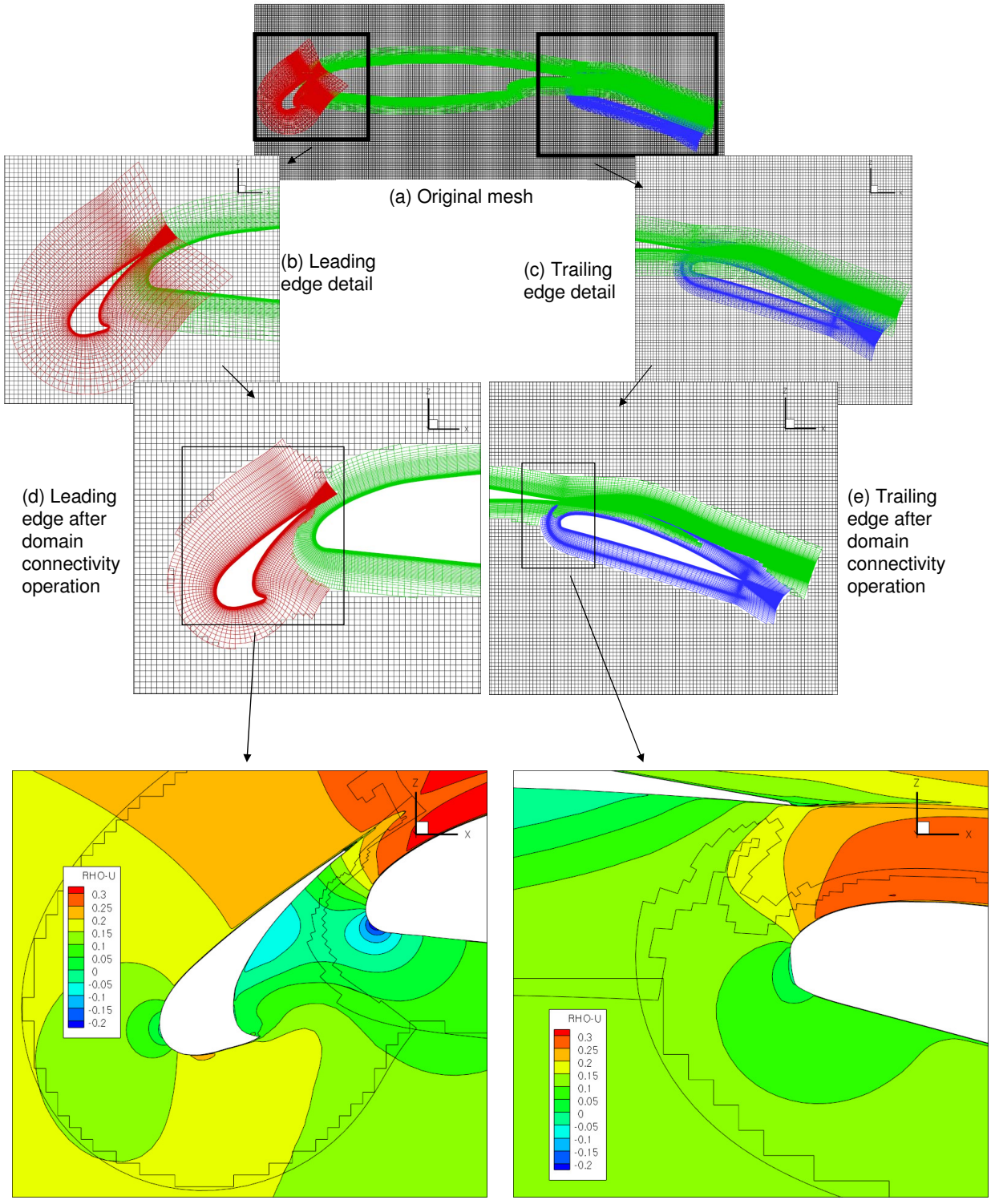


Figure 11. Prediction of Steady 2-D flow over NHLP-2D slotted airfoil at M=0.1937, α=4 deg and Re=3.52e6 using HELIOS; Comparision of surface pressure distributions to experimental data



(f) Contours of stream-wise momentum near the slat and the flap, continuity of contour lines at mesh interfaces indicates accuracy of domain connectivity and solution interpolation

Figure 12. domain-connectivity and flow solutions for NHLP airfoil-flap-slat test case using HELIOS; Conditions are M=0.197, α=4 deg, Re=3.52e6

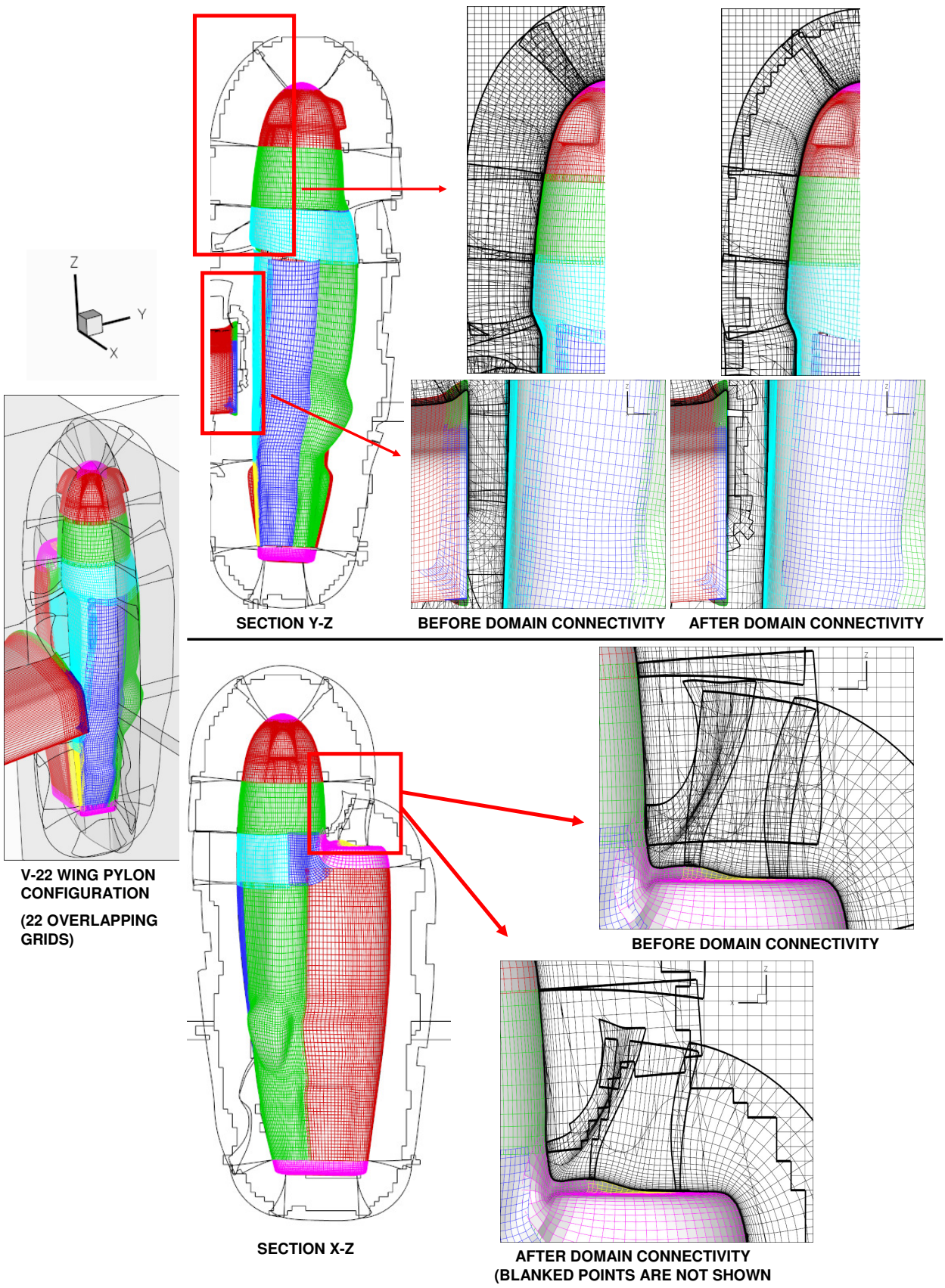


Figure 13. Mesh system and domain-connectivity solutions for the V-22 wing pylon geometry with 6.7 million grid points and 22 grid blocks using PUNDIT

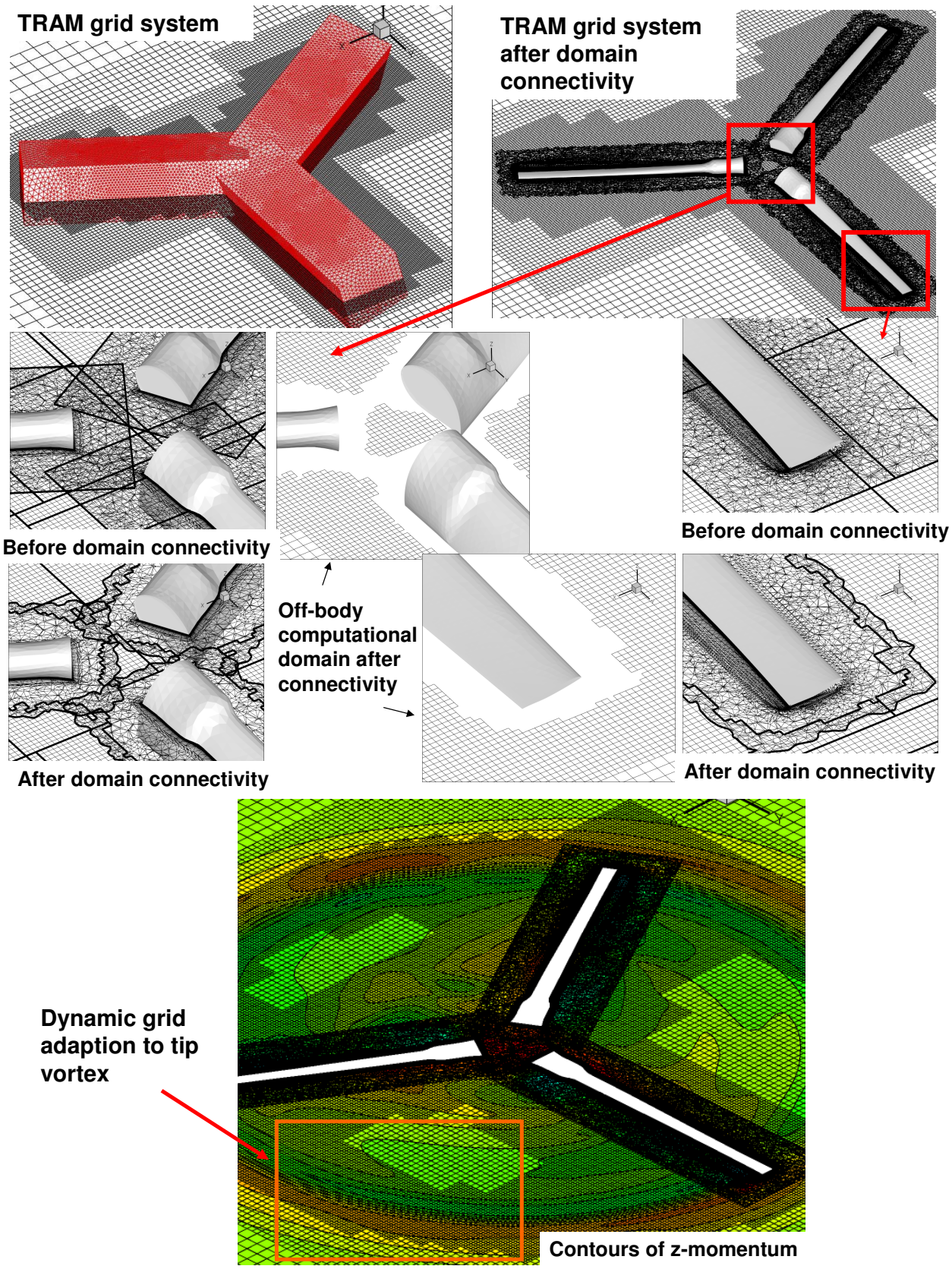
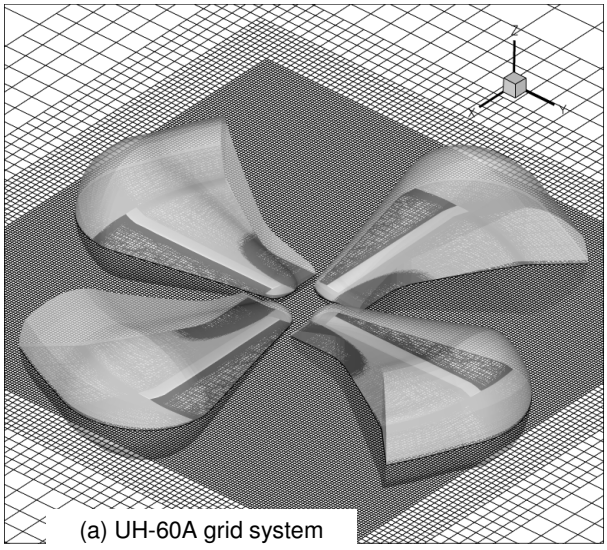
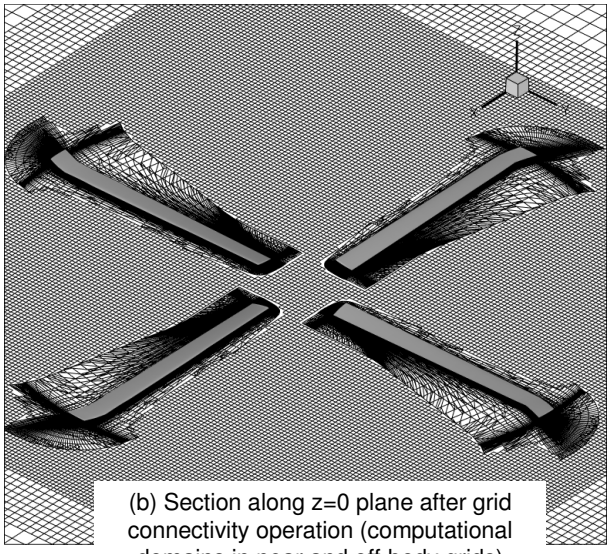


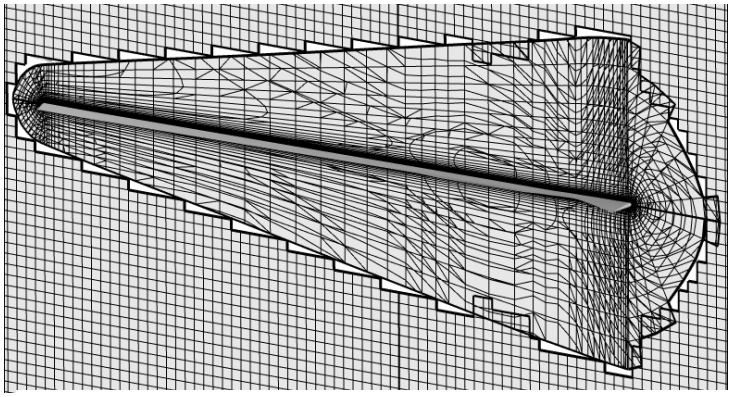
Figure 14. Mesh system, domain-connectivity and flow solutions for 1/4 scaled V-22 (TRAM) rotor using HELIOS; Plots illustrate grid overlap, hole geometry and dynamic mesh adaption to flow features.



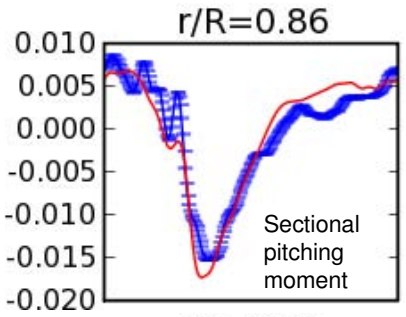
(a) UH-60A grid system

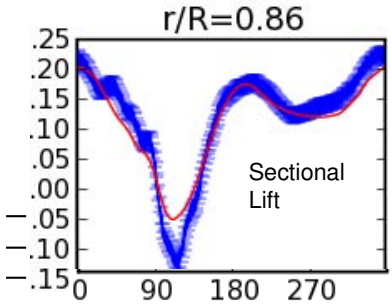


(b) Section along z=0 plane after grid connectivity operation (computational domains in near and off-body grids)

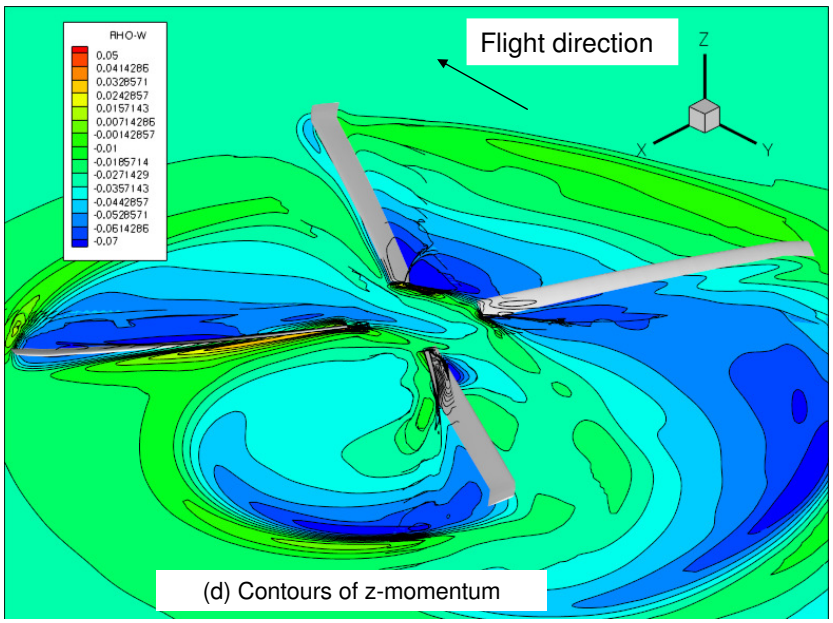


(c) Radial section along the rotor blade showing computational domains in near and off-body grids





(e) Comparison of predicted aerodynamic loading (red -) with test data (blue +-)



(d) Contours of z-momentum

Figure 15. Mesh system, domain-connectivity and flow solution for aeroelastic simulation of UH-60A rotor using HELIOS; Plots illustrate grid overlaps, hole geometry and comparison with measured flight test data.